Complexity Classification of Network Information Flow Problems

April Rasala Lehman* Eric Lehman†

Abstract

We address the *network information flow problem*, in which messages available to a set of sources must be passed through a network to a set of sinks with specified demands. This differs from traditional multicommodity flow, because information can be duplicated and encoded. Previous work has focused on the special case of multicasting using linear coding. In this paper, we explore the applicability of network coding to a breadth of problems and consider the greater potential of nonlinear coding techniques. Our main contribution is a taxonomy of network information flow problems. We establish a three-way partition consisting of problems solvable without resorting to network coding, problems requiring network coding that are polynomial-time solvable, and problems for which obtaining a linear network coding solution is NP-hard. We also demonstrate limitations of linear coding: for multicasting, nonlinear codes may employ a smaller alphabet than any linear code and, more generally, there exist solvable information flow problems that do not admit a linear solution.

1 Introduction

We address the *network information flow problem*. Here a network is a directed acyclic graph containing sources (each with a set of available messages), some intermediate nodes, and sinks (each demanding a set of messages). A message is a symbol drawn from an alphabet Σ, which is typically a finite field. One symbol may be sent across each edge. The problem is to determine whether the message demands at every sink can be satisfied. This resembles a multicommodity flow problem, but is fundamentally different because information can be duplicated and encoded, while commodities can not.

*MIT Laboratory for Computer Science. Cambridge, MA 02139. arasala@theory.lcs.mit.edu

†MIT Laboratory for Computer Science. Cambridge, MA 02139. e_lehman@theory.lcs.mit.edu

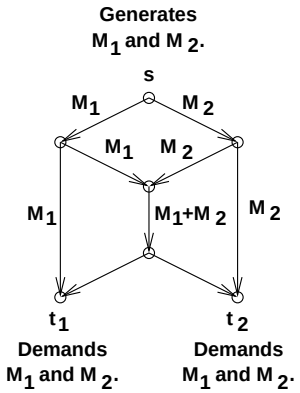

Figure 1: The canonical example of network coding.

Figure 1 shows an instance where information flow differs from commodity flow. The single source s has two messages, M_1 and M_2, which are both demanded by two sinks, t_1 and t_2. This problem can be solved only by coding. We send M_1 on the left, M_2 on the right, and $M_1 + M_2$ down the middle. Each sink can then compute M_1 and M_2 from the data it receives.

1.1 Previous Work The network information flow problem was introduced by Ahlswede, Cai, Li and Yeung [1]. They considered the *multicast* problem in which a single source broadcasts a number of messages to a set of sinks. Ahlswede, et al. showed that if a multicast problem with k messages is solvable, then the minimum cut between the source and each sink must be of size at least k. Li and Yeung [6] showed that this bound is tight and achievable by a restricted class of codes referred to as *linear network codes*. A linear network code has the property that the symbol transmitted by a node on an outgoing edge is a linear combination of the symbols received at that node. Koetter and Médard [5, 4] introduced an algebraic framework for study of the problem. They reduced the problem of determining whether an instance of the network information

flow problem admits a linear solution to determining whether a related polynomial lies in the ideal of a particular variety. They note that the usual approach to this problem employs Gröbner bases and can take exponential time in general. However, for the specific case of the multicast problem, their framework yields a randomized polynomial time algorithm. Ho, Karger, Médard, and Koetter reformulated these algebraic conditions in more combinatorial terms [3]. Recently, Sanders, Egner, and Tolhuizen [7], improving upon an exponential-time algorithm due to Li and Yeung [6], presented a deterministic, polynomial-time algorithm for the multicast problem. In addition, Sanders et al. showed an instance of the multicast problem where the rate achievable by network coding is a factor $\Omega(\log V)$ greater than the best possible without coding, where V is the number of nodes in the network.

1.2 Our Contribution In this paper, we explore the applicability and limitations of network coding to a breadth of information flow problems beyond multicast. Our main contribution is a taxonomy of information flow problems based on the number of sources, number of sinks, distribution of messages at the sources, and relationship between sink demands. We describe a three-way partition of possible information flow problems. For the first class, we prove that network coding adds nothing; if sink demands can be satisfied at all, traditional flow technique provide a solution in polynomial time that does not involve coding. Then we exhibit a second class of network information flow problems for which coding is advantageous. In this case, solutions can be obtained in polynomial time by adapting the Sanders-Li multicast algorithm. For the third class of problems, we show that determining whether there exists a solution using linear codes— as in the Sanders-Li algorithm— is NP-hard.

We also consider limits on the power of network coding. The Sanders-Li algorithm employs an alphabet Σ of size $O(m)$, where m is the number of sinks. We show that this can not be substantially improved; there exist multicast problems requiring an alphabet of size $\Omega(\sqrt{m})$. However, determining the minimum alphabet size for a specific multicast problem is NP-hard, whether we consider linear or more general codes. Moreover, we show that linear coding is limited in several respects. First, nonlinear codes may allow a significantly smaller alphabet, even for multicast problems. Furthermore, we show that there exist solvable information flow problems that do not admit any linear solution.

2 The Model

We model an instance of the *information flow problem* with a directed, acyclic graph $G = (V, E)$. There are two disjoint subsets of special nodes, $S, T \subseteq V$. The nodes in $S = \{s_1, \ldots, s_n\}$ are called *sources* and have indegree zero. The nodes in $T = \{t_1, \ldots, t_m\}$ are called *sinks* and have outdegree zero. All other nodes in V are called *intermediate nodes*.

The goal of the problem is to transmit a collection of *messages* $\mathcal{M} = \{M_1, M_2, \ldots\}$ from the sources to the sinks. Each message consists of one symbol drawn from an alphabet Σ of size $q > 1$. (We shall frequently regard Σ as a finite field.) Each source s_i has a set of available messages $\mathcal{A}(s_i) \subseteq \mathcal{M}$, and each sink t_i demands a set of messages $\mathcal{D}(t_i) \subseteq \mathcal{M}$.

One symbol from Σ can be transmitted over each edge in the directed graph G. In particular, for each source s_i, the symbol on the edge $s_i \to v$ must be a function of the messages available at the source, $\mathcal{A}(s_i)$. If u is an intermediate node, then the symbol on edge $u \to v$ must be a function of the symbols on the edges entering u.

The set of all functions associated with edges is called a *network code*. If the alphabet Σ is a field and every edge function is a linear combination of its inputs, then we say that the network code is *linear*. If every edge function maps one input directly to its output, then we say that the network code is *trivial*. If, for every sink t_i, every message demanded by t_i is computable as a function of the symbols on the incoming edges, then the network code is a *solution* to that instance of the information flow problem.

In practice, if we can transmit a collection of single-symbol messages $\mathcal{M}$ from the sources to the sinks, then we can equally well transmit multi-symbol data streams by sending one symbol from each stream in each time step.

Multicast information flow problems have received the most study to date. In this case, there is one source s, and there are multiple sinks $t_1, \ldots, t_m$. All messages are available at the source and demanded by every sink; that is, $\mathcal{A}(s) = \mathcal{M}$ and $\mathcal{D}(t_i) = \mathcal{M}$ for all $1 \leq i \leq m$.

3 The Role of Alphabet Size

The size q of the alphabet Σ used in a network code has an important practical interpretation. Each intermediate node receives one symbol from Σ along each incoming edge and then transmits functions of those symbols along its outgoing edges. Thus, in general, each node requires $\log q$ memory bits per incoming edge. Moreover, if each edge transmits a constant number of bits per time step, then use of a large alphabet adds $\log q$ time steps of latency per node. Consequently, we would like to keep the alphabet size q as small as possible.

3.1 A Lower Bound on Alphabet Size The Sanders-Li algorithm shows by construction that every solvable multicast problem can be solved using a linear network code with an alphabet of size $O(m)$, where m is the number of sinks. We show that this result is essentially tight; even if we restrict attention to multicast problems and allow nonlinear network codes, there exist solvable multicast problems requiring an alphabet of size $\Omega(\sqrt{m})$. The implication is that $\Theta(\log m)$ memory bits are necessary and sufficient to store a single alphabet symbol.

We begin with a technical lemma that gives some insight into why a larger alphabet may be necessary to solve a network coding problem. Let f_i and f_j be functions mapping Σ^2 to Σ. Then we can form a function $g_{ij} : \Sigma^2 \to \Sigma^2$ defined by:

$$g_{ij}(\alpha, \beta) \quad = \quad (\, f_i(\alpha, \beta) \, , \, f_j(\alpha, \beta) \,)$$

If g_{ij} is invertible, then we say that functions f_i and f_j are *independent*. Equivalently, f_i and f_j are independent if and only if there do not exist distinct points (α_1, β_1) and (α_2, β_2) in Σ^2 such that $f_i(\alpha_1, \beta_1) = f_i(\alpha_2, \beta_2)$ and $f_j(\alpha_1, \beta_1) = f_j(\alpha_2, \beta_2)$.

In short, the messages α and β can be determined from $f_i(\alpha, \beta)$ and $f_j(\alpha, \beta)$ if and only if the functions f_i and f_j are independent. But the following lemma says we can not construct a large set of pairwise independent functions over a small alphabet. The main idea of our subsequent theorems is that some information flow problems are insolvable with a small alphabet because one "runs out" of independent functions.

LEMMA 3.1. *If $f_1, \ldots, f_n$ are pairwise independent functions of the form $f_i : \Sigma^2 \to \Sigma$, then $n \leq q + 1$.*

Proof. First, we show that each function f_i must be a q-to-1 mapping. Suppose not. Then f_i must take on some value $\gamma \in \Sigma$ at more than q points $(\alpha, \beta) \in \Sigma^2$. By the pigeonhole principle, the function f_j (where $j \neq i$) must take on some value $\delta \in \Sigma$ for at least two of those points; call them (α_1, β_1) and (α_2, β_2). Thus, we have $f_i(\alpha_1, \beta_1) = f_i(\alpha_2, \beta_2)$ and $f_j(\alpha_1, \beta_1) = f_j(\alpha_2, \beta_2)$, contradicting the assumption that f_i and f_j are independent.

Now define an *agreement* of the function f_i to be a pair of distinct points (α_1, β_1) and (α_2, β_2) in Σ^2 such that $f_i(\alpha_1, \beta_1) = f_i(\alpha_2, \beta_2)$. Each function f_i has $q^2(q-1)/2$ agreements; for each of the q elements $\gamma \in \Sigma$, we can choose $q(q-1)/2$ pairs of distinct points from among the q points in $(\alpha, \beta) \in \Sigma^2$ such that $f_i(\alpha, \beta) = \gamma$. In all, there are $q^2(q^2-1)/2$ pairs of distinct points in Σ^2. Therefore, again by the pigeonhole principle, there must exist two different functions f_i and f_j that share an agreement if:

$$\begin{aligned} n \cdot q^2(q-1)/2 &> q^2(q^2-1)/2 \\ n &> q+1 \end{aligned}$$

But if f_i and f_j share an agreement (α_1, β_1) and (α_2, β_2), then we have $f_i(\alpha_1, \beta_1) = f_i(\alpha_2, \beta_2)$ and $f_j(\alpha_1, \beta_1) = f_j(\alpha_2, \beta_2)$, contradicting the assumption that f_i and f_j are independent. Therefore, we must have $n \leq q + 1$ as claimed. □

Conversely, constructing a set of $q + 1$ pairwise independent functions is a simple matter when q is a prime power. Regard Σ as a finite field, and take all functions $f(x, y)$ of the form $x + \alpha y$ where $\alpha \in \Sigma$ together with the function y.

THEOREM 3.1. *There exist solvable multicast information flow problems that require an alphabet of size $\Omega(\sqrt{m})$, even if nonlinear network codes are permitted.*

Proof. Define an information flow problem as follows. Create a single source s and intermediate nodes $v_1, \ldots, v_p$. Add a directed edge $s \to v_i$ from the source to each intermediate node. For each pair of distinct intermediate nodes v_i and v_j, create a sink t_{ij} and add directed edges $v_i \to t_{ij}$ and $v_j \to t_{ij}$. Note that the number of intermediate nodes p is $\Theta(\sqrt{m})$, where m is the number of sinks. There are two messages, M_1 and M_2 available at the source, and these two messages are demanded by every sink.

First, we show that this problem is solvable. If the alphabet size is a prime power greater than the number of intermediate nodes, then the edges $s \to v_1, \ldots, s \to v_p$ may carry functions of M_1 and M_2 that are pairwise independent. If each intermediate node then relays its input directly to its outputs, then each sink receives pairwise independent functions of M_1 and M_2 and can therefore reconstruct messages M_1 and M_2 as desired.

Now, we show that the problem is insolvable if the alphabet is too small. Suppose that the number of intermediate nodes is greater than $q+1$. Then by Lemma 3.1 the edges $s \to v_i$ can not carry functions of M_1 and M_2 that are all pairwise independent. In particular, suppose that intermediate nodes v_i and v_j receive functions that are not independent. Then messages M_1 and M_2 can not be determined from the values of these functions. Therefore, these messages can not be computed at sink t_{ij}, which receives no other information. Thus, in general, an alphabet of size $\Omega(\sqrt{m})$ is required to solve some multicast information flow problems. □

3.2 Determining the Minimum Alphabet Size We have seen that some information flow problems can only be solved by using a large alphabet. In this section, we show that it is computationally hard to determine exactly how large an alphabet is required.

The following two reductions rely on the hardness of graph coloring and a generalization of graph coloring called H-coloring. In both cases, we map an undirected graph $G' = (V', E')$ to a network information flow problem G as follows. The nodes of G consist of a single source s, an intermediate node v_i for each vertex $v'_i \in V'$, and a sink t_{ij} for each edge $(v'_i, v'_j) \in E'$. There is an edge $s \to v_i$ for each vertex $v'_i \in V'$, and there are edges $v_i \to t_{ij}$ and $v_j \to t_{ij}$ for each edge $(v'_i, v'_j) \in E'$. Two messages, M_1 and M_2 are available at the source, and these two messages are demanded by every sink.

THEOREM 3.2. *Deciding whether there exists a linear network code with alphabet size q for a multicast information flow problem is NP-hard when q is a prime power.*

Proof. We use a reduction from vertex coloring on undirected graphs.

Let $G' = (V', E')$ be an undirected graph. Construct a corresponding information flow graph G as described above.

We show that G' is $q+1$ colorable if and only if G is solved by a linear network code with an alphabet of size q. First, suppose that G' is $(q+1)$-colorable in order to show that this implies the existence of a linear network code that solves the information flow problem G'. Let $c(i) \in \{1, \ldots, q+1\}$ denote the color of vertex v'_i. As noted after the proof of Lemma 3.1, there exist pairwise independent functions $f_1, \ldots, f_{q+1}$ of the form $f_i : \Sigma^2 \to \Sigma$. Along each edge $s \to v_i$ and all edges $v_i \to t_{ij}$, send the symbol $f_{c_i}(M_1, M_2)$. Then each sink t_{ij} receives $f_{c(i)}(M_1, M_2)$ and $f_{c(j)}(M_1, M_2)$. Since colors on adjacent vertices are distinct, $c(i) \neq c(j)$, and so the functions $f_{c(i)}$ and $f_{c(j)}$ are independent. Thus, each sink can reconstruct messages M_1 and M_2 as desired.

Next, suppose that there exists a linear network coding solution to the information flow problem G with an alphabet of size q. We show that this implies that there exists a $q+1$ coloring of G'. Each edge $s \to v_i$ then carries a nonzero linear combination $\alpha M_1 + \beta M_2$. We can partition the set of all such linear combinations into $q+1$ equivalence classes; the nonzero multiples of $M_1 + \alpha M_2$ form one class for each $\alpha \in \Sigma$ and the nonzero multiples of M_2 form the remaining class. This places every pair of independent linear combinations into different classes. Assign each class a distinct color. Now assign vertex $v'_i \in V'$ the color of the class containing the function associated with edge $s \to v_i$. The endpoints of each edge $(v'_i, v'_j) \in E'$ are then colored differently, because the functions for edges $s \to v_i$ and $s \to v_j$ must be independent so that sink t_{ij} can reconstruct messages M_1 and M_2. Therefore, this gives a valid $q+1$ coloring of G'. □

3.3 Nonlinear Codes We now consider nonlinear codes and show that minimizing the alphabet size remains hard. We use a reduction from H-coloring. An H-coloring of an undirected graph G is a homomorphism $h : G \to H$ such that $h(v)$ and $h(u)$ are adjacent vertices of H if v and u are adjacent vertices of G. Hell and Nešetřil showed that H-coloring is NP-hard whenever H is not bipartite and is solvable in polynomial time if H is bipartite[2].

THEOREM 3.3. *Deciding if there exists a network code with alphabet size q for a multicast information flow problem is NP-hard when q is a prime power.*

Proof. Define a graph H as follows. The vertices are all the functions $f : \Sigma^2 \to \Sigma$. There is an edge between vertices f and g if they are independent functions. Note that H is not bipartite for all prime powers q, since there exists a set of three pairwise independent functions.

Let $G' = (V', E')$ be an arbitrary undirected graph. Construct the corresponding network information flow problem G. We show that G' is H-colorable if and only if there exists a solution to problem G over an alphabet of size q.

Supose that G' has an H-coloring, h. Then we can solve the information flow problem G by sending each vertex v_i the symbol $f(M_1, M_2)$, where $f = h(v_i')$. Each sink receives inputs from two vertices v_i and v_j such that v_i' and v_j' are adjacent in G'. This means that $h(v_i')$ and $h(v_j')$ are adjacent in H and are therefore independent functions. Thus, the sink can reconstruct messages M_1 and M_2.

On the other hand, suppose that there is a solution to the information flow problem G. Then we can construct an H-coloring h of the graph G' as follows. For each vertex v_i in G, let $h(v_i')$ be the function of M_1 and M_2 output by v_i. If vertices v_i' and v_j' are adjacent in G', then the corresponding vertices v_i and v_j in G share a sink. Since the sink can reconstruct M_1 and M_2, the functions $h(v_i')$ and $h(v_j')$ must be independent and thus adjacent in H. Therefore, h is a valid H-coloring of G'. □

Interestingly, some multicast problems can be solved with a smaller alphabet by using *nonlinear* codes. For example, let Σ be an alphabet of size 3. Let G' be a graph whose vertices are all functions $f : \Sigma^2 \to \Sigma$ and whose edges are all pairs of independent functions. Then consider the corresponding information flow problem G. This problem can be solved using alphabet Σ by sending $f(M_1, M_2)$ to the vertex corresponding to the function f. On the other hand, suppose that we want a linear solution. This requires coloring the vertices of G' using independent linear functions. A computation shows that the chromatic number of G' is 6, which implies an alphabet of size at least 5. We conjecture that there can be a very large gap between the absolute smallest alphabet size for a multicast problem and the smallest possible alphabet size using linear coding.

4 Complexity Classification

We now provide a taxonomy of network information flow problems. For our purposes, an information flow problem is defined by four attributes: single or multiple sources, single or multiple sinks, message distribution at sources, and message distribution at sinks. Thus a class of information flow problems is defined by a four-tuple (α, β, γ, δ), which is interpreted as follows:

- α is 1 if there is a single source and n if there are multiple sources.
- β is 1 if there is a single sink and m if there are multiple sinks.
- γ is I if all messages are available at every source, D if the sources have available disjoint sets of messages, and A if there are no specific guarantees about the availability of messages at sources. In the case of a single source γ is I.
- δ is I if every sink demands every message, D if sinks demand disjoint sets of messages, and A if there are no specific demand guarantees. In the case of a single source γ is I.

We show that each of the resulting classes of information flow problems falls into one of the following three categories.

Trivial codes suffice: The simplest problems can be solved with a trivial network code, one in which every edge carries an unencoded message.

Linear codes suffice: The next set consists of problems for which nontrivial network coding is sometimes necessary. For this class of problems, linear network coding always suffices, if a solution exists. Furthermore, a solution can be found in polynomial time by adapting the Sanders-Li algorithm.

Hard: Finally, the remaining problems sometimes require nontrivial network coding, but determining if a linear solution exists is NP-hard. For this last class of problems, there are instances that do not permit linear solutions but are solvable with nonlinear codes.

In the next three subsections, we justify this categorization of problems.

Problem difficulty	# of sources	# of sinks	Information at sources	Information at sinks
Trivial codes suffice	1 or n	1	I, D or A	I
	1 or n	m	I	D
Linear codes suffice	1 or n	m	I, D or A	I
Hard to find linear codes, may need nonlinear codes	1 or n	m	I	A
	n	m	D or A	D or A

4.1 Problems with Trivial Network Coding Solutions We prove that two classes of problems can be solved with trivial network codes, in which every edge carries an unencoded message. Solutions can be found in polynomial time with standard flow algorithms.

THEOREM 4.1. *An instance of information flow problem, (n, m, I, D), with multiple sources, multiple sinks and each message available at every source but requested by exactly one sink has a solution if and only if there is a trivial network coding solution.*

Proof. We show that such an instance can be solved by augmenting the associated network G and finding an appropriate flow. In particular, we add a super-source s^* and add $|\mathcal{M}|$ edges from s^* to each source s_i, creating a multigraph G'. We also add a super-sink t^*, and for each sink node t_i we add $|\mathcal{D}(t_i)|$ edges from t_i to t^*. In G' the original sources and sinks are now intermediate nodes and all messages are available only at the super source s^* and requested by only the super-sink t^*.

Clearly, if the original problem was solvable, then so is the new one. If the new problem is solvable, then the maximum flow from s^* to t^* must be at least $|\mathcal{M}|$ units; by a counting argument, we can not transmit $|\mathcal{M}|$ messages across a cut with capacity less than $|\mathcal{M}|$. This implies that there exist $|\mathcal{M}|$ edge-disjoint flow paths from s^* to t^*. Our construction ensures that every flow path traverses a former-source s_i and that exactly $|\mathcal{D}(t_j)|$ paths traverse each former-sink t_j. Therefore, in the original problem, each message can be routed from a source to the appropriate sink on a path that is edge-disjoint from the paths taken by all other messages. Consequently, no coding is necessary. □

Next we turn our attention to information flow problems with a single sink and show that regardless of the number of sources there always exists a trivial network coding solution whenever the instance is solvable.

THEOREM 4.2. *An instance of the information flow problem with a single sink has a solution if and only if there is a trivial network coding solution.*

Proof. Let $G = (V, E)$ be the graph representing the underlying network of with the single sink t. Create graph $G' = (V', E')$ by adding to G a super source s^*. In addition, for each message M_i, add a node, μ_i, an edge from s^* to μ_i and add an edge from node μ_i to s_i if $M_i \in \mathcal{A}(s_i)$.

Clearly, if the original problem was solvable, then so is the new one. If the new problem is solvable, then the maximum flow from s^* to t must be of size at least $|\mathcal{M}|$ units; by a counting argument, we can not transmit $|\mathcal{M}|$ messages across a cut with capacity less than $|\mathcal{M}|$. If the maximum flow from s^* to t is of size $|\mathcal{M}|$, then one unit of flow passes through each node μ_i. Therefore we can use the edge-disjoint paths from the maximum flow to route each message to t without using coding. □

4.2 Polynomial Time Solvable Linear Coding Problems Recently Sanders et. al [7] presented the first deterministic polynomial time algorithm for solving the multicast information flow problem. As Sanders et. al note, their algorithm can be seen as a fast implementation of an algorithm due to Li et. al [6]. We refer to these algorithms together as the Sanders-Li algorithm. This algorithm can be easily adapted to also solve the information flow problem with multiple sources provided that all sinks request to receive all messages.

The Sanders-Li algorithm was initially presented in terms of the multicast problem in which every sink wants to receive all available messages from a single source. The first step of the algorithm is to find a

flow of size k from the source to each sink. Let F_i be the flow associated with sink t_i. The edges are then considered in topological order. For each sink t_i, there are a set $\mathcal{E}(i)$ of $|\mathcal{M}|$ edges that are the last edge in each flow path of F_i considered by the algorithm. The Sanders-Li algorithm maintains the invariant that for each sink the set of symbols sent on edges in $\mathcal{E}(i)$ are linearly independent combinations of the messages.

THEOREM 4.3. *An instance of the information flow problem with every message requested by every sink is polynomial time solvable.*

*Proof.*Consider an instance of the information flow problem in which every message is requested by all the sinks. Let $G = (V, E)$ represent the associated network. Create a graph G' by adding a super source s^*. In addition, for each message M_i, add a node μ_i, an edge from s^* to μ_i and an edge $\mu_i \rightarrow s_j$ if $M_i \in \mathcal{A}(s_j)$.

For each sink t_i find a flow F_i of size $|\mathcal{M}|$ in G'. Using the corresponding portions of these flow paths in G and the Sanders-Li algorithm yields a linear network coding solution in polynomial time. □

Note that in the special case, (1, m, I, I), where there is only a single source the problem in which every sink requests every message corresponds to the multicast case and therefore the proof that it is polynomial time solvable is due to Sanders et al [7].

4.3 Hard Linear Network Coding Problems

We now consider the class of information flow problems (n, m, D, A), where there are multiple sources with disjoint information and multiple sinks that may demand arbitrary messages. We show that even determining whether there exists a linear network coding solution to such a problem is NP-hard. This contrasts with the information flow problems considered previously, for which linear solutions can be found efficiently, provided they exist. We focus on the class (n, m, D, A) for ease of presentation; similar arguments give hardness results for more restricted problem classes.

LEMMA 4.1. *Let f_1, f_2, f_3, h, and k be linear functions over a field. If x_1, x_2, and x_3 are uniquely determined by $f_1(x_1, z_1)$, $f_2(x_2, z_2)$, $f_3(x_3, z_3)$, $g(x_1, x_2, x_3, z_1, z_2, z_3)$ and $h(x_1, x_2, x_3, z_1, z_2, z_3)$ then $f_i(x_i, z_i) = \alpha x_i$ for some $i \in \{1, 2, 3\}$ and $\alpha \neq 0$.*

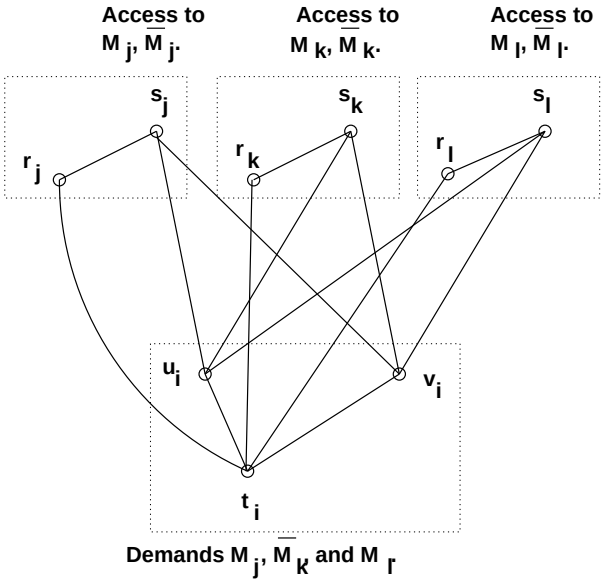

Figure 2: Portion of the information flow problem corresponding to variables x_j, x_k and x_l and the clause $c_i = (x_j \vee \overline{x_k} \vee x_l)$.

Proof. The values of the five functions can uniquely determine the values of at most five of the variables x_1, x_2, x_3, z_1, z_2 and z_3. If x_i is determined and $f_i(x_i, z_i)$ depends on z_i, then z_i is determined as well. Thus, at least one of the functions f_i does not depend on z_i, and so $f_i(x_i, z_i) = \alpha x_i$ as claimed. □

We now describe how to map a 3-CNF formula to an information flow problem in the class (n, m,D,A). Let ϕ be a 3-CNF formula over variables $x_1, x_2, \ldots x_z$. For each variable x_j in ϕ, we make the variable gadget shown in the top three boxes of Figure 4.3. This gadget consists of a source node s_j with access to messages two messages, M_j and $\overline{M_j}$. The source node has an outgoing edge to an intermediate node, r_j. For each clause $c_i = (x_j \vee \overline{x_k} \vee x_l)$, we create the clause gadget shown in the bottom box of Figure 4.3. This consists of a sink t_i, which demands messages M_j, $\overline{M_k}$, and M_l, together with two intermediate nodes, u_i and v_i. The variable gadget is connected to the clause gadget as follows. Nodes r_j, r_k, and r_l connect directly to the sink t_i. Nodes s_j, s_k, and s_l all connect to both u_i and v_i. This linkage is illustrated in Figure 4.3. (Note that all three variable gadets are connected to the clause gadget in the same way, even though variable x_k is negated in the clause. This negation is reflected in the demands at the sink.)

LEMMA 4.2. *A 3-CNF formula ϕ is satisfiable if and only if the corresponding information flow problem has a linear network coding solution.*

Proof. Suppose that ϕ is satisfied by some assigment π. If a variable x_j is true in π, then source s_j sends message M_j to r_j and sends message $\overline{M_j}$ on all other outgoing edges. If x_j is false, then s_j sends $\overline{M_j}$ to r_j and sends M_j on all other edges. Node r_j passes its input to its output. Now consider the gadget associated with a clause such as $c_i = (x_j \vee \overline{x_k} \vee x_l)$. Since π is a satisfying assignment, at least one literal in the clause is true and so at most two literals are false. Each message corresponding to a true literal is sent to the sink from an r-node. Each message corresponding to a false literal is sent from a source node to both u_i and v_i. Since there are at most two such messages, u_i and v_i can relay them both to the sink. (For example, suppose that assignment π makes x_j true and both $\overline{x_k}$ and x_l false. Then the sink receives message M_j via node r_j, message $\overline{M_k}$ via node u_i, and message M_l via node v_i.) Thus, all sink demands are satisfied, and the information flow problem has a linear solution.

In the other direction, suppose that there is a linear solution to the information flow problem. We construct an assignment π as follows. If the output of r_j is a function of only M_j, then set x_j true. If the output is a function of only $\overline{M_j}$, then set x_j false. Otherwise, set x_j arbitrarily. Now consider a clause $c_i = (x_j \vee \overline{x_k} \vee x_l)$. Let f_j, f_k, f_l denote the values output by r_j, r_k, and r_l, and let g and h denote the values output by u_i and v_i. Since the sink can determine messages M_j, $\overline{M_k}$, and M_l, Lemma 4.1 implies that either f_j depends only on M_j, f_k depends only on $\overline{M_k}$, or f_l depends only on M_l. Therefore, at least one of the literals x_j, $\overline{x_k}$, or x_l is true, and the clause is satisfied by assignment π. Therefore, π is a satisfying assignment for the 3-CNF ϕ. □

Lemma 4.2 establishes the hardess of information flow problems in the class (n, m, D, A). Minor adjustments to the reduction yield the following, more general result:

THEOREM 4.4. *Determining whether there exist linear network coding solutions to information flow problems in the classes (1, m, I, A), (n, m, D or A, D or A), and (n, m, I, A) is NP-hard.*

4.4 Linear Network Codes are Insufficient If we do not insist on a linear solution, then the information flow problems generated by 3-CNF formulas are *always* solvable— even if the 3-CNF formula was not satisfiable. This is in stark contrast to the multicast case where a linear solution exists if any solution exists.

THEOREM 4.5. *There are solvable information flow problems in the classes (1, m, I, A), (n, m, D or A, D or A), and (n, m, I, A) for which there is no linear network coding solution.*

Proof. [for the class (n, m, D, A)] Let ϕ be an unsatisfiable 3-CNF formula. By Lemma 4.2, there does not exist a linear solution to the corresponding information flow problem. However, we can construct a nonlinear network coding solution to this information flow problem as follows. Take the alphabet $\Sigma = \Gamma \times \Gamma$, where Γ is an arbitrary alphabet. Thus, each message is a pair of symbols drawn from Γ. Each source node s_j sends the first symbol of messages M_j and $\overline{M_j}$ to node r_j and sends the second symbol of these two messages on all other outgoing edges. Node r_j relays its input to its output. Now consider a clause $c_i = (x_j \vee \overline{x_k} \vee x_l)$. The sink s_i demands messages M_j, $\overline{M_k}$, and M_l. From nodes r_j, r_k, and r_l, the sink receives the first symbol of all six messages M_j, $\overline{M_j}$, M_k, $\overline{M_k}$, M_l, and $\overline{M_l}$. Furthermore, the nodes u_i and v_i receive the second symbol of all six messages. Since each of these nodes can send two symbols from Γ to the sink, they can together provide the sink with the second symbol of the three messages it demands. □

5 Open Problems

We have shown that there are network information flow problems for which linear codes are computationally hard to find and others for which linear codes do not even exist. Thus, exploring nonlinear network codes seems like a fruitful direction for future work. The ultimate goal is to determine necessary and sufficient conditions for the existence of a solution to an arbitrary network information flow problem.

We have seen that, even in the multicast case, a somewhat smaller alphabet can be used if one employs nonlinear coding rather than linear. We suspect that the gap can actually be very large, but this requires lower-bounding the chromatic number of graphs like the one constructed in Section 3.3. We

are unable to do this at present.

Acknowledgments

Madhu Sudan first introduced us to this problem and gave many helpful suggestions and comments. We would like to thank David Karger for pointing us to the work of Sanders et al. [7].

References

[1] Rudolf Ahlswede, Ning Cai, Shuo-Yen Robert Li, and Raymond W. Yeung. Network information flow. *IEEE Transactions on Information Theory*, 46(4):1204–1216, July 2000.

[2] Pavol Hell and Jaroslav Nešetřil. On the complexity of h-coloring. *Journal of Combinatorial Theory, Series B*, 48:92–110, 1990.

[3] Tracey Ho, David Karger, Muriel Médard, and Ralf Koetter. Network coding from a network flow perspective. In *IEEE International Symposium on Information Theory*, 2003.

[4] Ralf Koetter and Muriel Médard. Beyond routing: An algebraic approach to network coding. In *INFOCOM*, 2002.

[5] Ralf Koetter and Muriel Médard. An algebraic approach to network coding. *IEEE/ACM Transactions on Networking*, To Appear.

[6] Shuo-Yen Robert Li, Raymond W. Yeung, and Ning Cai. Linear network coding. *IEEE/ACM Transactions on Information Theory*, 49:371–381, 2003.

[7] Peter Sanders, Sebastian Egner, and Ludo Tolhuizen. Polynomial time algorithms for network information flow.